IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:	**Kyle G. Brown, Stephen G. Graham, Steven M. Miller and Mark D. Weitzel**	Group Art Unit:	**2145**
		Examiner:	**J.R. Swearingen**
Appln. No.	**10/014,106**	Confirmation No.	**2639**
Filed:	**December 11, 2001**	Attorney Docket No.	**RSW920010188US1**
Title:	**METHOD AND APPARATUS FOR DYNAMIC RECONFIGURATION OF WEB SERVICES INFRASTRUCTURE**		

FILED ELECTRONICALLY

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Attention: Board of Patent Appeals and Interferences

APPELLANTS' BRIEF

This brief is in furtherance of the Notice of Appeal filed in this case on or about November 16, 2007.

1) REQUIRED FEE

The requisite fee of $510.00 set forth in §41.20(b)(2) is submitted herewith. If the submitted fee is insufficient, the United States Patent and Trademark Office (hereinafter "Office") is authorized to charged Applicant's Deposit Account No. 09-0461 for any shortfall. A one-month extension of time and the fee of $120 is being filed concurrently herewith.

2) REAL PARTY IN INTEREST

The present application is assigned to International Business Machines Corporation. Accordingly, International Business Machines Corporation is the real party in interest.

3) RELATED APPEALS AND INTERFERENCES

The appellant, assignee, and the legal representatives of both are unaware of any other appeal or interference that will directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

4) STATUS OF CLAIMS

a. Claims canceled: None

b. Claims withdrawn from consideration but not canceled: None

c. Claims pending: 1-48

d. Claims allowed: None

e. Claims merely objected to: None

f. Claims rejected: 1-48

g. Claims appealed: 1-48

Appealed claims 1-48 as currently pending are attached as Appendix A hereto.

5) STATUS OF AMENDMENTS

There are no un-entered amendments to the specification claims or drawings in this case.

6) SUMMARY OF CLAIMED SUBJECT MATTER

The present invention relates to the maintenance, sharing, and movement of Web services in a network. As described on page 3, lines 11 et seq. of the present specification, “Web services” are application logic (or software modules) that can be exposed to and shared with other nodes over the Internet via a standardized interface mechanism.

The invention provides a software construct (termed a Web service “container” in the specification) for managing Web services at a network node and an adaptive model for the dynamic configuration for a plurality of Web service containers distributed throughout a network in a software and hardware platform-independent manner. The containers dynamically adapt themselves and, particularly, the Web services contained

therein, based on a pluggable set of heuristics. The containers can exchange Web services software on a peer-to-peer basis or through querying a registry. The containers can discover the Web services contained in other containers. The present invention allows containers to dynamically exchange Web services software as well as contextual information, such as current workload, so that containers are virtually limitlessly reconfigurable based on context. For instance, containers may, not only reconfigure themselves as routers to other containers containing requested Web services, but also load and unload Web service software modules based on detected workload and/or Web service availability at other servers and send software modules to peer containers as needed to allow them to run a Web service locally rather than from a remote location on the network.

The invention also enables one server with a Web services container to send Web services software to another server with a Web services container in order to allow that other server to begin providing that service. This may be useful, for instance, when the work load at a first server exceeds that server's capabilities. That server can then send the Web service software to one or more other servers and then divide the servicing of requests for that service among two or more servers. The first server can reconfigure itself either partially or totally as a "service router" to route requests for given Web services to other servers that it has determined can provide that service either by virtue of it having itself sent the Web service software to the other server(s) or by querying the other server(s) as to the contents of their Web service containers.

Another exemplary routing scenario involves a request being received by the first server. Acting as a "service router", the first server directs the request to the second server. When the second server responds to the request, it includes context information that indicates it was the node that ultimately handled the request. The container that initiated the request understands the contextual information returned and, for each subsequent request, directs the outgoing message to the second server. This can be logically thought of as dynamically adding a WSDL port to the service and indicating that the new port is the preferred endpoint. In this manner, any number of "hops" may be taken before reaching the ultimate destination, but network efficiency is gained by

providing a mechanism that avoids unnecessary processing from the second service request forward.

Not only can Web services software be exchanged in a platform independent manner, but the Web service containers themselves are platform independent. That is, the Web service containers at two different network nodes can be implemented in different programming languages and run on different platforms, while still being able to exchange contextual information and Web service software modules using SOAP and WSDL.

Below is a table mapping the claim recitations to the portion of the specification and/or drawing Figures that disclose the recited subject matter.

Claim Recitation	**Support in Specification and/or Drawings**
1. A computer program product (hereinafter "container") recorded on computer readable medium for organizing and manipulating Web services software modules on a network, comprising:	Page 10, lines 12-15
computer executable instructions for determining and describing Web services software modules that are available at a corresponding, local network node,	Page 13, line 19-page 14, line 6
said Web services software modules comprising executable software modules that can be exchanged between nodes of a network and run at said nodes;	Page 3, lines 11-13
computer executable instructions for generating messages to be transmitted to other containers via a network disclosing said Web services software modules that are available at said corresponding network node, and including contextual information about said container and said Web services available at said corresponding, local node;	Page 13, line 19- page 14, line 6 and Figure 3, 315 thru 325
computer executable instructions for receiving and deciphering messages disclosing Web services software modules that are available at other network nodes corresponding to other containers; and	Page 14:10-17 and Figure 3, 327-331
computer executable instructions for causing the dynamic reconfiguration of said Web services software modules available at said corresponding network node on said network based on said transmitted and received messages, including the exchange of said Web services software modules between said network nodes;	Page 18, line 14- page 19, line 5
wherein said container is in the form of a Web services	Page 10, line 15- page

software module.	11, line 7
2. The computer program product of Claim 1 wherein said computer executable instructions for causing the dynamic reconfiguration of Web services software modules comprises: computer executable instructions for transmitting messages to said other containers requesting said other containers to return copies of Web services software modules; and computer executable instructions, responsive to receipt of messages from said other containers requesting copies of Web services software modules available at said corresponding network node, for sending copies of said requested Web services software modules to said requesting containers.	Page 24, lines 4-18 and Figure 4C
3. The computer program product of Claim 2 wherein said computer executable instructions for transmitting messages generates messages that are hardware and software platform independent.	Page 8, lines 10-15 and page 11, lines 1-7
4. The computer program product of Claim 3 wherein said computer executable instructions for transmitting messages further comprises computer executable instructions for transmitting said messages to and from a Web services registry and said computer executable instructions for receiving and deciphering messages further comprises computer readable code for receiving said messages from a Web service registry.	Page 12, line 4-13, line 6 and Figure 2
5. The computer program product of Claim 4 wherein said messages disclosing said Web services software modules that are available at network nodes are in the Web Services Descriptor Language (WSDL).	Page 8, lines 11-15 and page 10, lines 15-16
6. The computer program product of Claim 5 wherein said registry is a Universal Description, Discovery, and Integration initiative (UDDI) registry.	Page 11, lines 8-15 and 12, line 4-13, line 6
7. The computer program product of Claim 3 wherein said computer executable instructions for transmitting messages	Page 11, lines 8-18 and page 18, lines 14-20

uses a peer to peer messaging protocol between said containers and said computer executable instructions for receiving and deciphering messages uses a peer to peer messaging protocol between containers.	
8. The computer program product of Claim 7 wherein said messaging protocol is SOAP.	Page 11, lines 18-20 and page 13, line 7-15, line 5 and Figure 3
9. The computer program product of Claim 8 wherein said disclosures of said Web services software modules that are available at network nodes are contained in headers of Simple Object Access Protocol (SOAP) messages.	Page 11, lines 18-20 and page 13, line 7-15, line 5 and Figure 3
10. The computer program product of Claim 7 wherein said messaging protocol is JXTA.	Page 11, lines 18-20
11. The computer program product of claim 3 further comprising: computer executable instructions for receiving client requests for use of a Web services software module from client computers via said network.	Page 21, line 1-22, line 8, esp. page 21, lines 9-11 and Figure 4A, 226a
12. The computer program product of claim 11 wherein said computer executable instructions for causing the dynamic reconfiguration of Web services software modules comprises: computer executable instructions that, responsive to receipt of one of said client requests from a client for a Web services software module that is not available at said corresponding network node; determines, based on said received messages disclosing said Web services software modules that are available at other network nodes, whether another network node has a copy of said particular Web services software module; and invokes a proxy to another of said containers having a copy of a particular Web services software module based on said determination.	Page 21, line 1- page 22, line 8, and Figures 4A and 4B

13. The computer program product of claim 12 wherein said proxy comprises: computer executable instructions for routing said client requests for a Web services software module that is not available at said corresponding network node and has been determined to be available at another network node to another container corresponding to said another network node; computer executable instructions for receiving responses to said client requests from said another network node; and computer executable instructions for returning said responses to said requesting clients.	Page 21, line 1- page 22, line 8, and Figures 4A and 4B
14. The computer program product of claim 13 further comprising: computer executable instructions for receiving said client requests routed from another of said containers and causing said client requests to be handled by a copy of said particular Web services software module at a network node corresponding to said container to generate said response; and computer executable instructions for transmitting said response to said another container that routed said client request to said container.	Page 21, line 1- page 22, line 8, and Figures 4A and 4B
15. The computer program product of claim 11 further comprising: computer executable instructions for determining a load of client requests at said corresponding network node; and said computer executable instructions for causing the dynamic reconfiguration of Web services software modules performs said dynamic reconfiguration based on said load determination.	34, line 12- page 35, line 21 and Figures 4F, 4G, and 4H

16. The computer program product of claim 15 wherein said computer executable instructions for causing the dynamic reconfiguration of Web services software modules further comprises: computer executable instructions that, responsive to determination of a load of client requests for a particular Web services software module that is not available at said corresponding network node exceeding a predetermined level, issues a message requesting a copy of said particular Web services software module from another container that has a copy of said particular Web services software module; computer executable instructions for receiving and locally invoking said particular Web services software module from said other container; and computer executable instructions for routing client requests for said particular Web services software module to said local invocation of said particular Web services software modules.	Page 34, line 12- page 35, line 21 and Figures 4F, 4G, and 4H
17. The computer program product of claim 16 wherein said computer executable instructions for causing the dynamic reconfiguration of Web services software modules further comprises: computer executable instructions for offloading said particular Web services software module received from said other container responsive to said load of client requests for said particular Web services software module dropping below a second predetermined level.	Page 27, lines 1-5 and Figure 6

18. The computer program product of claim 15 wherein said computer executable instructions for causing the dynamic reconfiguration of Web services software modules comprises: computer executable instructions that, responsive to determination of a load of client requests for a particular Web services software module available at said corresponding network node exceeding a predetermined level, issues a message requesting another container to accept a copy of the code of said particular Web software modules from said computer program product; and computer executable instructions for sending a copy of said code of said particular Web services software module to said other container responsive to affirmative responses to said message requesting another container to accept a copy of the code of said particular Web services software module from said computer program product.	Page 27, line 17- page 30, line 10 and Figure 7
19. The computer program product of claim 18 wherein said computer executable instructions for causing the dynamic reconfiguration of Web services software modules further comprises: computer executable instructions for reconfiguring said computer program product to route client requests for said particular Web services software module to said other container.	Page 27, line 17- page 30, line 10 and Figure 7
20. The computer program product of claim 19 wherein said other container comprises a plurality of other containers.	Page 27, line 17- page 30, line 10 and Figure 7
21. The computer program product of claim 20 wherein said computer executable instructions for reconfiguring said computer program product to route client requests for said particular Web services software module to said other container distributes said client requests for said particular Web services software module between said other containers and said local invocation of said particular Web services software module.	Page 27, line 17- page 30, line 10 and Figure 7
22. The computer program product of claim 11 wherein said client requests indicate whether said requesting client has a container and a platform on which said client is running and wherein said computer program product further comprises computer executable instructions to read said client requests to determine whether said client has a container and said	Page 30, line 11- page 31, line 12 and Figure 4D

platform.	
23. The computer program product of claim 22 wherein said computer executable instructions for causing the dynamic reconfiguration of Web services software modules further comprising: computer executable instructions for sending a copy of the code of a particular Web services software module responsive to a client request for said Web services software module.	Page 31, lines 18-20
24. The computer program product of claim 11 further comprising: computer executable instructions for monitoring usage of Web services software modules by clients; and computer executable instructions for charging said clients for said usage.	Page 33, lines 1-16
25. A method for organizing and manipulating Web services software modules, comprising the steps of: providing a computer program product (hereinafter "container") at each of a plurality of said network nodes; each said container:	Page 10, lines 12-15
(1) determining and describing Web services software modules that are available at a corresponding network node,	Page 13, line 19- page 14, line 6
said Web services software modules comprising executable software modules that can be exchanged between nodes of a network and run at said nodes;	Page 3, lines 11-13
(2) transmitting messages via a network disclosing said Web services software modules that are available at said corresponding network node to other network nodes via said network and including contextual information about said container and said Web services available at said corresponding, local node;	Page 13, line 19- page 14, line 6 and Figure 3, 315 thru 325
(3) receiving and deciphering messages from other network nodes disclosing Web services software modules that are available at other network nodes; and	Page 14:10-17 Figure 3, 327-331
(4) dynamically reconfiguring Web services software modules on said network node transmitted and received based on said messages, including the exchange of said Web services software modules between said network nodes;	Page 18, line 14- page 19, line 5
wherein said method is in the form of a Web services software module.	Page 10, line 15- page 11, line 7

26. The method of Claim 25 wherein step (4) further comprises: (4.1) transmitting messages to said other network nodes requesting said other network nodes to return copies of Web services software modules; and (4.2) computer executable instructions, responsive to receipt of messages from said other network nodes requesting copies of Web services software modules, for sending said requested Web services software modules to said requesting network node.	Page 24, lines 4-18 and Figure 4C
27. The method of Claim 26 wherein said messages of steps (2) and (3) are hardware and software platform independent.	Page 8, lines 10-15 and page 11, lines 1-7
28. The method of Claim 27 wherein steps (2) and (3) comprise sending and receiving said messages to and from a Web services registry.	Page 12, line 4- page 13, line 6 and Figure 2
29. The method of Claim 28 wherein said messages disclosing said Web services software modules that are available at network nodes are in the Web Services Descriptor Language (WSDL).	Page 8, line 11-15 and page 10, line 15-16
30. The method of Claim 29 wherein said registry is a Universal Description, Discovery, and Integration initiative (UDDI) registry.	Page 11, lines 8-15 and page 12, line 4-13, line 6
31. The method of Claim 27 wherein steps (2) and (3) comprise sending and receiving said messages using a peer to peer messaging protocol between said network nodes.	Page 11, lines 8-18 and page 18, lines 14-20
32 The method of Claim 31 wherein said messaging protocol is SOAP.	Page 11, lines 18-20 and page 13, line 7- page 15, line 5 and Figure 3
33. The method of Claim 32 wherein said disclosures of said Web services software modules that are available at network nodes are contained in headers of Simple Object Access Protocol (SOAP) messages.	Page 11, lines 18-20 and page 13, line 7-15, line 5 and Figure 3
34 The method of Claim 31 herein said messaging protocol is JXTA.	Page 11, lines 18-20
35. The method of claim 27 further comprising the step of:	Page 21, line 1- page 22,

(5) receiving client requests for Web services software modules from client computers via said network.	line 8, esp. page 21, lines 9-11 and Figure 4A, 226a
36. The method of claim 35 wherein step (4) further comprises: (4.3) responsive to receipt of a client request from a client for a Web services software module that is not available at said corresponding network node; determining, based on said received messages disclosing said Web services software modules that are available at network nodes, what network nodes have copies of said particular Web services software module; and invoking a proxy to another of said network nodes having a copy of a particular Web services software module based on said determination.	Page 21, line 1- page 22, line 8, and Figures 4A and 4B
37. The method of claim 36 wherein said proxy performs the steps of: (4.3.1) routing client requests for said particular Web services software module to said other of said network nodes; (4.3.2) receiving responses to said client requests; and (4.3.3) returning said responses to said requesting clients.	Page 21, line 1- page 22, line 8, and Figures 4A and 4B
38. The method of claim 37 further comprising: (6) receiving said client requests forwarded from other of said network nodes and causing said client requests to be handled by said copy of said particular Web services software module corresponding to said network node to generate said response; and (7) transmitting said response to said network node that issued said client request.	Page 21, line 1- page 22, line 8, and Figures 4A and 4B
39. The method of claim 35 further comprising: (8) determining a load of client requests at said corresponding network node; and wherein, in step (4), said dynamic reconfiguration is performed based on said load determination.	Page 34, line 12- page 35, line 21 and Figures 4F, 4G, and 4H

40. The method of claim 39 herein step (4) further comprises: (4.4) responsive to determination of a load of client requests for a particular Web services software module that is not available at said corresponding network node exceeding a predetermined level, issuing a message requesting a copy of the code of said particular Web services software module from another network node that has a copy of said particular Web services software module; (4.5) receiving and locally invoking said code for said particular Web services software module from said other network node; and (4.6) routing client requests for said particular Web services software module to said local invocation of said code for said particular Web services software module.	Page 34, line 12- page 35, line 21 and Figures 4F, 4G, and 4H
41. The method of claim 40 wherein step (4) further comprises: (4.7) offloading said local code for said particular Web services software module responsive to said load of client requests for said particular Web services software module dropping below a second predetermined level.	Page 27, lines 1-5 and Figure 6
42. The method of claim 39 wherein step (4) comprises: (4.8) responsive to determination of a load of client requests for a particular Web services software module available at said corresponding network node exceeding a predetermined level, issuing a message requesting another network node to accept a copy of the code of said particular Web services software module from said network node; and (4.9) sending a copy of said code of said particular Web services software module to said other network node responsive to affirmative responses to said message requesting another network node to accept a copy of the code of said particular Web services software module from said network node.	Page 27, line 17- page 30, line 10 and Figure 7
43. The method of claim 42 wherein step (4) further comprises: (4.10) reconfiguring said network node to route client requests for said particular Web services software module to said other network node.	Page 27, line 17- page 30, line 10 and Figure 7
44. The method of claim 43 wherein said other network node comprises a plurality of other network nodes.	Page 27, line 17- page 30, line 10 and Figure 7

45. The method of claim 44 wherein step (4.10) comprises distributing said client requests for said particular Web services software module between said other network nodes and said local invocation of said particular Web services software module.	Page 27, line 17- page 30, line 10 and Figure 7
46. The method of claim 35 wherein said client requests indicate a platform on which said client is running and wherein said method further comprises the step of: (9) reading said client requests to determine said platform of said client.	Page 30, line 11- page 31, line 12 and Figure 4D
47. The method of claim 46 further comprising the step of: (10) sending a copy of the code of a particular Web services software module responsive to a client request for said Web services software module.	Page 31, lines 18-20
48. The method of claim 35 further comprising the steps of: (11) monitoring usage of Web services software modules by clients; and (12) charging said clients for said usage.	Page 33, lines 1-16
49. The method of claim 1 wherein said contextual information includes at least one of an identity of a Web service, the capabilities of said Web service, the operating system of said Web service, the platform of said Web service, the Web services hosted by a container type Web service, the workload of said Web service, and a network location of said Web service.	Page 6, line 20- page 7, line 3, page 13, line 20- page 14, line 2, page 15, line 6-10, and page 30, line 11- page 31, line 12

7) GROUNDS FOR REJECTION TO BE REVIEWED ON APPEAL

1. Claims 1-4, 7, 11-28, 31, and 35-48 stand rejected under 35 U.S.C. §102(e) as anticipated by U.S. Published Patent Application No. 2002/0111814 issued to Barnett et al. (hereinafter Barnett).

2. Claims 5, 6, 8, 9, 29, 30, 32, and 33 stand rejected under 35 U.S.C. '103(a) as unpatentable over Barnett in view of an article by Sycara (hereinafter Sycara).

3. Claims 10 and 34 stand rejected under 35 U.S.C. 103(a) as unpatentable over Barnett in view of Project JXTA.

8) ARGUMENT

The sole or primary reference in all of the rejections is Barnett. However, Barnett pertains to the fundamentally different problem of client-server sharing of Web services (whereas the present invention is directed to server-server configuration of and sharing of Web services) and, in any event, teaches a completely different paradigm for handling such Web services.

I. The Barnett Reference

Barnett pertains to a distributed computing platform architecture for facilitating dynamic availability of e-commerce services over a network. Barnett describes a system in which client machines access Web services available at network servers. Discovery by the clients of what Web services are available to it is done by consulting with a particular Web server 200 that maintains a database of the available Web services. Abstract of Barnett.

Particularly, a client that wants to know what Web services are available to it logs on to server 200 and provides the server with its group identification. Server 200 maintains a database indicating what Web services are available to members of the different groups and provides that information to the client. The Web services that are available to the clients are not on the server 200 but are on different, look-up servers 230. Paragraphs [0031], [0033], and [0035]-[0036]. The client machines, server 200, and the look up servers 230 (with which the Web services are associated) all have different software for interacting with the other types of nodes on the network.

This is a completely different paradigm than that of the present invention. First, Barnett discusses client machines accessing and using Web services found on servers and does not relate to the discovery, exchange and dynamic reconfiguration of Web services between and amongst a plurality of peer servers. Second, one of the key features of the present invention is that a plurality of servers on a network have the same piece of software, i.e., the "container", for carrying out the discovery and exchanging of Web services, whereas the Examiner's rejection is based on an amalgamation of pieces of software at the client machine, server 200, and look up servers 230, each of which contains different software. Third, the container itself is a

Web service, which is not disclosed in Barnett, even under the Examiner's definition of Web service and what is a container in Barnett.

In Barnett, a client having the client software consults server 200 and LDAP 220 to determine what Web services are available to the client from the look up servers 230. However, Barnett does not disclose server 200 determining what Web services it has locally, or downloading Web services locally, or the other servers 230 determining what Web services are available at other servers. Barnett does not discuss how servers 230 obtain copies of the Web services. Barnett is concerned with a different issue, namely, clients using the Web services found on the servers. Barnett, on the other hand, addresses client use of web services and has nothing to do with the dynamic reconfiguration and maintenance of web services software modules amongst the servers. The present invention has almost nothing to do with the accessing and use of those web services software modules by the actual clients

Each of the three different types of nodes on the network discussed in Barnett (clients, server 200, and servers 230) has different software for performing different functions particular to that type of node. This is very different than the present invention, which is primarily a peer to peer paradigm in which all of the servers have similar "container" software and manage, disclose, and discover what Web services they themselves have and what Web services are available at other nodes of the network, and permit exchanging of Web services, serving as proxy nodes for Web services, etc.

Thus, Barnett discloses a completely different paradigm than the present invention. The present invention pertains to a method and apparatus for dynamically reconfiguring Web services software modules amongst a plurality of different servers on the web in order to facilitate their use by client machines over the web

II. Discussion

A. Independent Claim 1 Patentably Distinguishes Over Barnett

The independent claims, claims 1 and 25 clearly recite these distinctions over Barnett.

1. Barnett Does Not Meet the First Element of Claim 1

Referring first to claim 1 for instance, claim 1 recites that the container is located at a particular network node and comprises instructions for "determining and describing web services software modules that are available at a corresponding, local network node". The Examiner asserts that this is found in Barnett in paragraph [0038]. However, paragraph [0038] discusses LoadBalancer 270 on server 200 and particularly says that LoadBalancer 270 "maintains a list of registries (look up servers not expressly shown in 270 of Fig. 2) it finds when it registers itself".

Thus, LoadBalancer 270 maintains a list of look up servers 230, (registries being the same thing as look up servers) and it is the list of registries that is not expressly shown in 270 of Figure 2 (since some registries/look up servers 270 are, in fact, shown in Figure 2). No other reading of paragraph [0038] would make sense. Certainly, Barnett cannot mean that the LoadBalancer 270 has within it look up servers having web services. A server inside of a LoadBalancer running on another server would be nonsense. Yet this appears to be the position that the Examiner is taking. What is inside LoadBalancer 270 is a list of look up servers, not the look up servers themselves. Not only would it make no sense for there to be look up servers in LoadBalancer 270, but Figure 2 and the rest of the specification (particularly paragraph [0036] discussed below) make clear that the Web services are available through the look up servers 230 and that the look up servers are separate nodes on the network. Particularly, Figure 2 shows look up servers 230 and they are not connected to the LoadBalancer 270 (except through the network, of course). What it does not show is the list of look up servers in LoadBalancer 270 as described in paragraph [0038]. Furthermore, paragraph [0038] is preceded by paragraph [0036], which describes in detail how a user's browser 205 accesses the server 200 to determine what services 240 are available to it, such as specific database 250, which Figure 2 shows as being available through one of the look up servers 230, not through the LoadBalancer 270 itself.

Accordingly, contrary to the Examiner's assertions, Barnett does not disclose the first element of claim 1 of "determining and describing web services software modules that are available at a corresponding, local network node".

2. Barnett Doe Not Meet the Second Element of Claim 1

Barnett also does not meet the requirements of the second element of claim 1.

The second element of claim 1 is instructions "for generating messages to be transmitted to other containers via a network disclosing said web services software modules that are available at said corresponding network node". The Examiner asserts that this is found in paragraph [0036] of Barnett. However, as mentioned above, paragraph [0036] discloses how a first server node, namely, web server node 200, provides information to the client browser node 205 (not to "other containers" as claimed, i.e., another server node) as to the web services that are available at other server nodes, namely, look up servers 230 (not at the "corresponding network node" as claimed).

In response to this argument as made by Applicant during prosecution, the Examiner responded that "it is unclear what Applicant's traversal is in reference to". It appears that the Examiner believes that Applicant's argument did not address the specific claim language allegedly lacking from Barnett.

It is apparent from the above-noted matters that the Examiner did not understand Applicant's argument. Applicant's argument was and still is that:

> (1) according to the express language of claim 1, all of the elements recited in claim 1 are in a single piece of software, i.e., the container;
> (2) according to the express language of claim 1, the container is located at a single server node; and
> (3) according to the express language of claim 1, the web services software modules that are "contained" in the container reside at that same single server node.

As described above in detail, in Barnett, the software that the Examiner is relying on as allegedly teaching the software for generating the messages disclosing the available web services is at a different node than the web services. Therefore, Barnett cannot meet these limitations.

3. Barnett Does Not Meet the Last Element of Claim 1

Barnett also does not meet the recitation in the last paragraph of claim 1 that the container itself is a Web service.

The Examiner asserted that Barnett discloses in paragraph [0037] that its software is a Web service also. However, paragraph [0037] merely discloses that the ExecutionManager software module is a servlet. The fact that ExecutionManager is a servlet does not make it a Web service. Paragraph [0037] discloses:

> [0037] ExecutionManager 260 is a servlet which provides a mechanism for all services to run. In particular, the ExecutionManager 260 listens for communications from the client applet and if it reads a valid Executable, the ExecutionManager passes it to a LoadBalancer/ComputeServer(s) 270. The ExecutionManager 260 also uses the LookupFinder class to find the available LoadBalancers to execute computationally intensive jobs. ExecutionManager also utilizes a ServiceFinder class to find ComputeServers. If there is an error reading the Executable, an error message is returned to the client applet. When the ExecutionManager has read a valid Executable, it attempts to pass it to the LoadBalancer/ComputeServers (270 of FIG. 2).

This does not meet the definition of Web services expressly set forth in claim 1, namely, "executable software modules that can be exchanged between nodes of a network and run at said nodes".

With respect to Applicant's arguments on this subject, the Examiner asserted that:

> A container cannot be a web service, since a web service is a method of performing a function on the web. This statement is contrary to Applicant's own claims that state a container is a web services software module.

Applicant does not understand the Examiner's position. The Examiner seems to be saying that Applicant has admitted that a container cannot be a web service by virtue of having stated that a container is a web services software module. The term "web service" is simply a shortened version of "web services software module". They are one and the same thing in Applicant's application.

4. General Distinctions of Claim 1 Over Barnett

a. The Examiner Is Relying on Multiple Pieces Of Software at Different Network Nodes for Teaching The Elements of a Single Container of the Present Invention

In addition to all of the above, there is no single piece of software in Barnett that performs all the functions recited in claim 1 for a single container. Specifically, claim 1 recites that the container software module (1) determines the Web services available locally, (2) discloses to other nodes the Web services available at the local node, (3) receives and deciphers messages from other nodes to discover what Web services are available at those other network nodes, and (4) can dynamically reconfigure Web services available at its node based on messages exchanged between the nodes.

There is no single piece of software in Barnett that can perform all of these functions because clients 205, server 200, and servers 230 located at different nodes of the network all have different software modules that perform different functions. No single node has software that performs all of the functions recited in claim 1 for the "container".

This is a completely different paradigm. One of the primary points of the present invention is that all of the nodes have a "container" that can perform all of the aforementioned functions and that the container itself is a Web service. Barnett teaches a completely different paradigm in which clients, servers 230, and server 200 all perform different functions. Also, nothing in Barnett suggests that the software modules that perform the functions that the Examiner relies on are Web service software modules themselves.

b. Barnett Does Not Disclose Exchanging Contextual Information Between Containers

Finally, claim 1 recites that the messages transmitted between containers also include contextual information. As described in the specification, contextual information comprises information about the context of the Web service, such as the capabilities of the Web service, the operating system of the Web service, the platform of the Web

service, the Web services hosted by a container, the workload of the Web service, and the network location of the Web service.

Barnett does not disclose the transmission of any contextual information about the services. In Barnett, for instance, a separate LoadBalancer 270 handles such tasks as determining the most eligible Compute Server. Paragraphs [0038]-[0040]. Contextual information in Barnett is handled within the Execution Manager 260 and/or the LoadBalancer 270.

B. Independent Claim 25 Patentably Distinguish Over Barnett

Claim 25 recites essentially the same subject matter discussed above as claim 1, but as a method rather than a computer product claim. Particularly, claim 25 recites at least the following distinctions over Barnett: (1) "providing a computer program product (hereinafter "container") at each of a plurality of said network nodes"; (2) "determining and describing Web services software modules that are available at a corresponding network node"; (3) "transmitting messages via the network disclosing said Web services software modules that are available at said corresponding node to other network nodes via said network"; (4) the messages "including contextual information about said container and said Web services available at said corresponding, local node, (5) "receiving and deciphering messages from other network nodes disclosing Web services software modules that are available at other network nodes"; (6) "dynamically reconfiguring Web services software modules on said network node transmitted and received based on said messages"; and (7) "wherein said method is embodied in a Web services software module".

In summary, the container of the present invention is a software module that manages the life cycle of Web services available at the corresponding network node and these are the types of functions set forth in independent claims 1 and 25 as discussed above. Barnett's Execution manager, on the other hand "is a servlet that allows a constant point of entry for the client applet. The ExecutionManager reads in an executable object from the client applet and passes it to a LoadBalancer. The LoadBalancer is found through a Lookup Server". Barnett, paragraph [0101].

The ExecutionManager does not manage the life cycle of the Web services, rather it is just an entry point into the Web services.

Accordingly, independent claims 1 and 25 distinguish over the prior art of record.

C. The Dependent Claims Add Even Further Distinguishing Features

The dependent claims distinguish over the prior art of record for at least all of the same reasons set forth above with respect to the independent claims. The secondary references cited in connection with the obviousness rejections do not overcome the shortcomings of the primary reference Barnett discussed above. Accordingly, the obviousness rejections also have been overcome.

Furthermore, the dependent claims add even further patentably distinguishing features.

1. Dependent Claims 7 and 31

Dependent claim 7 further distinguishes over the prior art of record by reciting that "said computer executable instructions for transmitting messages uses a peer to peer messaging protocol between said containers and said computer executable instructions for receiving and deciphering messages uses a peer to peer messaging protocol between containers". Dependent claim 31 contains very similar recitations. The Examiner asserted that Barnett teaches these recitations, but, unlike the other claim rejections, did not cite a portion of Barnett to support this proposition. Applicant cannot find a portion of Barnett that supports this proposition.

2. Claims 12-20, 36-38, and 40-45

Many of the dependent claims are directed to very specific examples of dynamic reconfiguration of web services software modules between two (or more) servers. These include claims 12-20 depending directly or indirectly from claim 1 and claims 36-38, 40-45 depending directly or indirectly from claim 25.

Each of these claims specifically recites or incorporates a specific scenario in which web services software modules are dynamically reconfigured between two or more servers. In some of the examples, web services software modules are sent over

the network to other servers. In other scenarios, a server acts as a proxy for another server without actually copying, moving, or sending the web services software module over the network to another server.

All of these claims clearly patentably distinguish over Barnett, which does not discuss anything even remotely resembling proxying or moving web services software modules between two or more servers.

The Examiner relies on one or more of paragraphs [0035], [0037], [0039], and [0049] with respect to all of these claims. For convenience, Applicant has reproduced paragraphs [0035] through [0040] and paragraph [0049] of Barnett below.

> [0035] The client applet is structured so that services 240 to which the user has access automatically appear on the UI screen. These services are dynamic. If a new service is created and started while the user is working, and the user has permission to access that service, it will appear in the client applet. Conversely, if a service 240 becomes unavailable during the user's session, access to that service will disappear from that client applet only if it is a "remote" service. Local services have the ability to be completely downloaded to the client so they don't need to go away from the UI if they vanish from the server. This dynamic feature is one of the primary benefits of using the eCommerce Software Platform of the present invention as the infrastructure for eCommerce services.
>
> [0036] As mentioned, when the user via browser 205 gains access to the web server 200 after logging in, group information is provided to the client applet from the LDAP 220 Lookup Server(s) 230 provide an interface between users allowed to access group information thereby determining which services 240 are available. For example, the client may wish to access or utilize a specific database 250 registered with Lookup server 230.
>
> [0037] ExecutionManager 260 is a servlet which provides a mechanism for all services to run. In particular, the ExecutionManager 260 listens for communications from the client applet and if it reads a valid Executable, the ExecutionManager passes it to a LoadBalancer/ComputeServer(s) 270. The ExecutionManager 260 also uses the LookupFinder class to find the available LoadBalancers to execute computationally intensive jobs. ExecutionManager also utilizes a ServiceFinder class to find ComputeServers. If there is an error reading the Executable, an error message is returned to the client applet. When the ExecutionManager has read a valid Executable, it attempts to pass it to the LoadBalancer/ComputeServers (270 of FIG. 2).

> [0038] More particularly, LoadBalancer 270 maintains a list of registries (lookup servers not expressly shown in 270 of FIG. 2) it finds when it registers itself, and uses all available ComputeServers available to a LoadBalancer for a job. Information about the owner of the Executable is provided by other functionality 280 and stored via a "Result" object in a Result archive 285. For example, a ResultsManager object may combine the information ("Result") with the job ID to create a unique path (URL) to the Result, sent to the user via e-mail.
>
> [0039] The LoadBalancer/ComputeServer 270 passes the job to the most eligible ComputeServer (not shown in 270 of FIG. 2), and identifies the ComputeServer to pass the next job to based on the total amount of time each ComputeServer of the distributed system has spent on the jobs, or based on the last time a computeServer completed a job (these are configurable options).
>
> [0040] If no LoadBalancer 270 is available, the ExecutionManager 260 returns a diagnostic message to the client applet, otherwise it invokes the LoadBalancer's run job method with the Executable as a parameter. If the LoadBalancer 270 run job is successfully passed to the LoadBalancer/ComputeServer(s) 270, the job is given a unique job I.D. that is returned to the client applet and passed along with the Executable for identification purposes.
>
> [0049] Two different types of services may be implemented by the eCommerce Software Platform architecture of this invention. The types are different from an implementation perspective. One: any service sub-classed from LocalService functions by downloading the entire object needed for execution of the service. Hence, no portion of the service is resident on any server. Such condition implies fairly limited functionality for the service because it cannot access service specific databases, or perform any remote processing."

These paragraphs merely mention "load balancing", but provide no detail whatsoever about how it is done. Thus, even if one were to improperly speculate as to the nature of the load balancing mentioned in Barnett, one can only assume that it is traditional load balancing in which the client requests are distributed amongst a plurality of servers based on load balancing. The client is informed of which server has been assigned to service it and then the client communicates with that server. This is completely different than what is claimed in claims 12-20, 36-38, and 40-45 in which the

Web service are distributed amongst servers based on load balancing. These claims do not have anything to do with traditional load balancing.

Turning to the claims, specifically, Claims 16 and 40 recite a dynamic reconfiguration scheme in which, when the load of requests for a Web service that is not available at a server exceeds a predetermined level, the corresponding container sends out a request to another container at another node that has a copy of the Web service asking for a copy of the Web service. The Examiner asserted that this is found in paragraph [0039] of Barnett. However, paragraph [0039] discusses a LoadBalancer/ComputeServer 270 that passes jobs to the most eligible ComputeServer based on various criteria. However, it does not discuss the loading of the Web services software modules themselves. Paragraph [0039] pertains strictly to traditional load balancing of client requests for Web services, not exchanging of Web services between nodes.The only discussion anywhere in Barnett of moving web services software modules is the conventional situation where some or all of the software comprising a web service might be transmitted to the client machine. However, this has nothing to do with what is being claimed in these claims.

Furthermore, claims 12-14 and 36-38, which relate to one server acting as a proxy for another server (i.e., taking a client request, forwarding it to another server, receiving a response from the other server, and forwarding it to the requesting client), concern quite a different process than Barnett's load balancing. While proxying by one container for the web services software modules in another container in one sense achieves a similar result as traditional client-request load balancing in that requests are distributed to a container according to load, "load balancing" *per se* is understood in the trade as being something quite different than what is claimed in these claims. Particularly, traditional load balancing deals with directing clients to send client requests to different servers in a server farm in order to balance the load of request handling at a web site. The proxying claimed in these claims, on the other hand, is a technique wherein the client machine continues to send requests to one particular server and is entirely unaware of the fact that a different server than the one it is communicating with is actually providing the Web service that it is using.

More specifically, dependent claims 12 and 36 recite that, responsive to receipt of a client request for a Web service that is not available at the corresponding node, the container determines whether another node has a copy of the particular Web service and invokes a proxy to the container at the other node. The Examiner asserted that this is found in paragraph [0035] of Barnett. However, paragraph [0035] does not disclose the dynamic reconfiguration of Web services at all. It only discusses the discovery of new Web services (or disappearance of old Web services). It does not discuss any actual dynamic reconfiguration and it certainly does not contain any discussion of invoking proxies to other containers.

Most of the remaining dependent claims recite with specificity of the types of dynamic reconfigurations that the container can carry out. Each of these claims clearly adds further distinguishing features over the prior art insofar as Barnett does not perform any dynamic reconfiguration of Web services amongst a plurality of servers (even accepting, arguendo, the Examiner's broad definition of Web services as encompassing Barnett's Web services), Barnett is concerned with the use of services by client machines. The present application does not pertain to the use of Web services by clients, but to the management of Web services amongst a plurality of servers.

Specifically, dependent claims 13 and 37 depend from claims 12 and 36, respectively, and recite that the proxy routes the client requests for a Web service that is not available at the corresponding node to the other node and receives the responses from the other node and forwards them to the requesting client. The Examiner asserted that that this is found also in the same paragraph [0035] of Barnett. However, as discussed above, paragraph [0035], not only does not disclose one container passing requests for a Web service and responses from a Web service to another node, it does not even pertain to the operation of Web services, but only to the discovery of Web services.

Dependent claims 14 and 38 depend from claims 13 and 37, respectively, and recite the steps performed by the container that receives a request from the proxy container. The Examiner cited paragraph [0049] of Barnett for allegedly teaching this feature. However, paragraph [0049] simply discusses a client machine downloading a Web service from a server. This has nothing to do with proxying. In the Final Office

Action, the Examiner asserted that claims 14 and 38 do not state any use of proxying. This is clearly incorrect. These claims depend from claims 13 and 37, respectively, which the Examiner does not dispute pertain to proxying, and recite the specific actions performed by the servicing server in response to the proxy server's requests.

Dependent claims at 17 and 41 depend from claims 16 and 40, respectively, and recite the feature of the node offloading the Web service after the load for that Web service drops below a predetermined level. The Examiner again cites paragraph [0039] of Barnett as disclosing this feature. However, as noted above, paragraph [0039] has nothing to do with this.

Claims 18, 19, 42, and 43 recite a different dynamic reconfiguration and load balancing scheme in which, when the load of requests for a particular Web service at a particular node exceeds a threshold, the container at that node asks the container at another node to accept a copy of the Web service from it, sends a copy of the Web service to the other node, and then routes requests that it receives for that Web service to the other node. The Examiner asserts that paragraphs [0037] and [0039] of Barnett disclose this scheme. However, as previously noted, these paragraphs of Barnett do not discuss rerouting of requests or copying or exchanging of Web services. These paragraphs discuss only load balancing of requests among a plurality of servers that already have copies of the Web service.

III. Conclusion

For the foregoing reasons, Applicant respectfully requests the Office to reverse the Examiner's rejection of Claims 1-48.

Respectfully submitted,

Dated: February 5, 2008

/Theodore Naccarella/

Theodore Naccarella, Reg. No. 33,023
Synnestvedt & Lechner LLP
2600 Aramark Tower; Suite 2600
Philadelphia, PA 19107

Telephone: (215) 923-4466
Facsimile: (215) 923-2189

TXN:pmf
S:\I\IBM\IBM Raleigh RSW\Patents\P25391 USA\PTO\Appeal Brief(3).doc

APPENDIX A: CLAIMS INVOLVED IN THIS APPEAL

1. A computer program product (hereinafter "container") recorded on computer readable medium for organizing and manipulating Web services software modules on a network, comprising:

computer executable instructions for determining and describing Web services software modules that are available at a corresponding, local network node, said Web services software modules comprising executable software modules that can be exchanged between nodes of a network and run at said nodes;

computer executable instructions for generating messages to be transmitted to other containers via a network disclosing said Web services software modules that are available at said corresponding network node, and including contextual information about said container and said Web services available at said corresponding, local node;

computer executable instructions for receiving and deciphering messages disclosing Web services software modules that are available at other network nodes corresponding to other containers; and

computer executable instructions for causing the dynamic reconfiguration of said Web services software modules available at said corresponding network node on said network based on said transmitted and received messages, including the exchange of said Web services software modules between said network nodes;

wherein said container is in the form of a Web services software module.

2. The computer program product of Claim 1 wherein said computer executable instructions for causing the dynamic reconfiguration of Web services software modules comprises:

computer executable instructions for transmitting messages to said other containers requesting said other containers to return copies of Web services software modules; and

computer executable instructions, responsive to receipt of messages from said other containers requesting copies of Web services software modules available at said

corresponding network node, for sending copies of said requested Web services software modules to said requesting containers.

3. The computer program product of Claim 2 wherein said computer executable instructions for transmitting messages generates messages that are hardware and software platform independent.

4. The computer program product of Claim 3 wherein said computer executable instructions for transmitting messages further comprises computer executable instructions for transmitting said messages to and from a Web services registry and said computer executable instructions for receiving and deciphering messages further comprises computer readable code for receiving said messages from a Web service registry.

5. The computer program product of Claim 4 wherein said messages disclosing said Web services software modules that are available at network nodes are in the Web Services Descriptor Language (WSDL).

6. The computer program product of Claim 5 wherein said registry is a Universal Description, Discovery, and Integration initiative (UDDI) registry.

7. The computer program product of Claim 3 wherein said computer executable instructions for transmitting messages uses a peer to peer messaging protocol between said containers and said computer executable instructions for receiving and deciphering messages uses a peer to peer messaging protocol between containers.

8. The computer program product of Claim 7 wherein said messaging protocol is SOAP.

9. The computer program product of Claim 8 wherein said disclosures of said Web services software modules that are available at network nodes are contained in headers of Simple Object Access Protocol (SOAP) messages.

10. The computer program product of Claim 7 wherein said messaging protocol is JXTA.

11. The computer program product of claim 3 further comprising:

computer executable instructions for receiving client requests for use of a Web services software module from client computers via said network.

12. The computer program product of claim 11 wherein said computer executable instructions for causing the dynamic reconfiguration of Web services software modules comprises:

computer executable instructions that, responsive to receipt of one of said client requests from a client for a Web services software module that is not available at said corresponding network node;

determines, based on said received messages disclosing said Web services software modules that are available at other network nodes, whether another network node has a copy of said particular Web services software module; and

invokes a proxy to another of said containers having a copy of a particular Web services software module based on said determination.

13. The computer program product of claim 12 wherein said proxy comprises:

computer executable instructions for routing said client requests for a Web services software module that is not available at said corresponding network node and has been determined to be available at another network node to another container corresponding to said another network node;

computer executable instructions for receiving responses to said client requests from said another network node; and

computer executable instructions for returning said responses to said requesting clients.

14. The computer program product of claim 13 further comprising:

computer executable instructions for receiving said client requests routed from another of said containers and causing said client requests to be handled by a copy of said particular Web services software module at a network node corresponding to said container to generate said response; and

computer executable instructions for transmitting said response to said another container that routed said client request to said container.

15. The computer program product of claim 11 further comprising:

computer executable instructions for determining a load of client requests at said corresponding network node; and

wherein said computer executable instructions for causing the dynamic reconfiguration of Web services software modules performs said dynamic reconfiguration based on said load determination.

16. The computer program product of claim 15 wherein said computer executable instructions for causing the dynamic reconfiguration of Web services software modules further comprises:

computer executable instructions that, responsive to determination of a load of client requests for a particular Web services software module that is not available at said corresponding network node exceeding a predetermined level, issues a message requesting a copy of said particular Web services software module from another container that has a copy of said particular Web services software module;

computer executable instructions for receiving and locally invoking said particular Web services software module from said other container; and

computer executable instructions for routing client requests for said particular Web services software module to said local invocation of said particular Web services software modules.

17. The computer program product of claim 16 wherein said computer executable instructions for causing the dynamic reconfiguration of Web services software modules further comprises:

computer executable instructions for offloading said particular Web services software module received from said other container responsive to said load of client requests for said particular Web services software module dropping below a second predetermined level.

18. The computer program product of claim 15 wherein said computer executable instructions for causing the dynamic reconfiguration of Web services software modules comprises:

computer executable instructions that, responsive to determination of a load of client requests for a particular Web services software module available at said corresponding network node exceeding a predetermined level, issues a message requesting another container to accept a copy of the code of said particular Web software modules from said computer program product; and

computer executable instructions for sending a copy of said code of said particular Web services software module to said other container responsive to affirmative responses to said message requesting another container to accept a copy of the code of said particular Web services software module from said computer program product.

19. The computer program product of claim 18 wherein said computer executable instructions for causing the dynamic reconfiguration of Web services software modules further comprises:

computer executable instructions for reconfiguring said computer program product to route client requests for said particular Web services software module to said other container.

20. The computer program product of claim 19 wherein said other container comprises a plurality of other containers.

21. The computer program product of claim 20 wherein said computer executable instructions for reconfiguring said computer program product to route client requests for said particular Web services software module to said other container distributes said client requests for said particular Web services software module between said other containers and said local invocation of said particular Web services software module.

22. The computer program product of claim 11 wherein said client requests indicate whether said requesting client has a container and a platform on which said client is running and wherein said computer program product further comprises computer executable instructions to read said client requests to determine whether said client has a container and said platform.

23. The computer program product of claim 22 wherein said computer executable instructions for causing the dynamic reconfiguration of Web services software modules further comprising:

computer executable instructions for sending a copy of the code of a particular Web services software module responsive to a client request for said Web services software module.

24. The computer program product of claim 11 further comprising:

computer executable instructions for monitoring usage of Web services software modules by clients; and

computer executable instructions for charging said clients for said usage.

25. A method for organizing and manipulating Web services software modules, comprising the steps of:

providing a computer program product (hereinafter "container") at each of a plurality of said network nodes; each said container:

(1) determining and describing Web services software modules that are available at a corresponding network node, said Web services software modules comprising executable software modules that can be exchanged between nodes of a network and run at said nodes;

(2) transmitting messages via a network disclosing said Web services software modules that are available at said corresponding network node to other network nodes via said network and including contextual information about said container and said Web services available at said corresponding, local node;

(3) receiving and deciphering messages from other network nodes disclosing Web services software modules that are available at other network nodes; and

(4) dynamically reconfiguring Web services software modules on said network node transmitted and received based on said messages, including the exchange of said Web services software modules between said network nodes; wherein said method is in the form of a Web services software module.

26. The method of Claim 25 wherein step (4) further comprises:

(4.1) transmitting messages to said other network nodes requesting said other network nodes to return copies of Web services software modules; and

(4.2) computer executable instructions, responsive to receipt of messages from said other network nodes requesting copies of Web services software modules, for sending said requested Web services software modules to said requesting network node.

27. The method of Claim 26 wherein said messages of steps (2) and (3) are hardware and software platform independent.

28. The method of Claim 27 wherein steps (2) and (3) comprise sending and receiving said messages to and from a Web services registry.

29. The method of Claim 28 wherein said messages disclosing said Web services software modules that are available at network nodes are in the Web Services Descriptor Language (WSDL).

30. The method of Claim 29 wherein said registry is a Universal Description, Discovery, and Integration initiative (UDDI) registry.

31. The method of Claim 27 herein steps (2) and (3) comprise sending and receiving said messages using a peer to peer messaging protocol between said network nodes.

32 The method of Claim 31 wherein said messaging protocol is SOAP.

33 The method of Claim 32 wherein said disclosures of said Web services software modules that are available at network nodes are contained in headers of Simple Object Access Protocol (SOAP) messages.

34 The method of Claim 31 herein said messaging protocol is JXTA.

35. The method of claim 27 further comprising the step of:

(5) receiving client requests for Web services software modules from client computers via said network.

36. The method of claim 35 wherein step (4) further comprises:

(4.3) responsive to receipt of a client request from a client for a Web services software module that is not available at said corresponding network node;

determining, based on said received messages disclosing said Web services software modules that are available at network nodes, what network nodes have copies of said particular Web services software module; and

invoking a proxy to another of said network nodes having a copy of a particular Web services software module based on said determination.

37. The method of claim 36 wherein said proxy performs the steps of:

(4.3.1) routing client requests for said particular Web services software module to said other of said network nodes;

(4.3.2) receiving responses to said client requests; and

(4.3.3) returning said responses to said requesting clients.

38. The method of claim 37 further comprising:

(6) receiving said client requests forwarded from other of said network nodes and causing said client requests to be handled by said copy of said particular Web services software module corresponding to said network node to generate said response; and

(7) transmitting said response to said network node that issued said client request.

39. The method of claim 35 further comprising:

(8) determining a load of client requests at said corresponding network node; and

wherein, in step (4), said dynamic reconfiguration is performed based on said load determination.

40. The method of claim 39 herein step (4) further comprises:

(4.4) responsive to determination of a load of client requests for a particular Web services software module that is not available at said corresponding network node exceeding a predetermined level, issuing a message requesting a copy of the code of said particular Web services software module from another network node that has a copy of said particular Web services software module;

(4.5) receiving and locally invoking said code for said particular Web services software module from said other network node; and

(4.6) routing client requests for said particular Web services software module to said local invocation of said code for said particular Web services software module.

41. The method of claim 40 wherein step (4) further comprises:

(4.7) offloading said local code for said particular Web services software module responsive to said load of client requests for said particular Web services software module dropping below a second predetermined level.

42. The method of claim 39 wherein step (4) comprises:

(4.8) responsive to determination of a load of client requests for a particular Web services software module available at said corresponding network node exceeding a predetermined level, issuing a message requesting another network node to accept a copy of the code of said particular Web services software module from said network node; and

(4.9) sending a copy of said code of said particular Web services software module to said other network node responsive to affirmative responses to said message requesting another network node to accept a copy of the code of said particular Web services software module from said network node.

43. The method of claim 42 wherein step (4) further comprises:

(4.10) reconfiguring said network node to route client requests for said particular Web services software module to said other network node.

44. The method of claim 43 wherein said other network node comprises a plurality of other network nodes.

45. The method of claim 44 wherein step (4.10) comprises distributing said client requests for said particular Web services software module between said other

network nodes and said local invocation of said particular Web services software module.

46. The method of claim 35 wherein said client requests indicate a platform on which said client is running and wherein said method further comprises the step of:

(9) reading said client requests to determine said platform of said client.

47. The method of claim 46 further comprising the step of:

(10) sending a copy of the code of a particular Web services software module responsive to a client request for said Web services software module.

48. The method of claim 35 further comprising the steps of:

(11) monitoring usage of Web services software modules by clients; and

(12) charging said clients for said usage.

49. The method of claim 1 wherein said contextual information includes at least one of an identity of a Web service, the capabilities of said Web service, the operating system of said Web service, the platform of said Web service, the Web services hosted by a container type Web service, the workload of said Web service, and a network location of said Web service.

EVIDENCE APPENDIX

None

RELATED PROCEEDINGS APPENDIX

None